L Number	Hits	Search Text	DB	Time stamp
1	7865	xml! OR (extensible! ADJ3 (markup! OR mark-up! OR "mark up" OR html!))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/07 09:34
2	200	(xml! OR (extensible! ADJ3 (markup! OR mark-up! OR "mark up" OR html!))) SAME (custom! OR customiz$5) SAME (database$2 OR "data base" OR metadata! OR "meta data")	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/07 09:35
3	24	((xml! OR (extensible! ADJ3 (markup! OR mark-up! OR "mark up" OR html!))) SAME (custom! OR customiz$5) SAME (database$2 OR "data base" OR metadata! OR "meta data")) AND ("search engine" OR "search engines") NOT @AD>20001230	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/07 09:36

02285402 SUPPLIER NUMBER: 54299494 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Server-Side JavaScript. (Technology Tutorial) (Column) (Tutorial)
Stanek, William Robert
PC Magazine, 231(1)
May 4, 1999
DOCUMENT TYPE: Column Tutorial ISSN: 0888-8507 LANGUAGE: English
RECORD TYPE: Fulltext; Abstract
WORD COUNT: 3163 LINE COUNT: 00256

... should have the prefix X-).

Database Access

Server-side JavaScript provides a complete Web-to- **database** solution, and whether you want to **create** a **database** - **query** **engine** or a complete **database** -management **system** , you can use server-side JavaScript's LiveWire **Database** Service to help get the job done. The service supports native drivers for DB2, Informix, Oracle, and Sybase, as well as for **databases** that conform to the ODBC standard. Features supported by the service include **database** pooling, pass-through SQL, cursors, and stored procedures.

Keep in mind though that your server...

02285402 SUPPLIER NUMBER: 54299494 (THIS IS THE FULL TEXT)
Server-Side JavaScript.(Technology Tutorial)(Column)(Tutorial)
Stanek, William Robert
PC Magazine, 231(1)
May 4, 1999
DOCUMENT TYPE: Column Tutorial ISSN: 0888-8507 LANGUAGE: English
RECORD TYPE: Fulltext; Abstract
WORD COUNT: 3163 LINE COUNT: 00256

ABSTRACT: Netscape's powerful server-side JavaScript technology is among the most popular programming languages in history. Running JavaScript on the server allows sophisticated Web-to-database connectivity. Client-side scripting can generate dynamic content and handle preliminary validation of user-input, but E-commerce usually requires a server component to parse data and handle follow-on processing. Server-side JavaScript handles these tasks more easily and effectively than traditional CGI, providing a well-developed framework for common applications. It is oriented toward three-tier client/server architectures and has Session Management, File System and Mail services. JavaScript's LiveWire Database Service supports native drivers for numerous databases. Four basic steps for working with databases are mapping the structure, establishing a connection, handling transactions and releasing the connection. An external library framework lets JavaScript support existing functions written in other programming languages. Netscape's LiveConnect is a set of Java extensions that lets JavaScript users go beyond the external library.

TEXT:

Corporate Web sites aren't novelties any more. More and more, these sites are viewed as critically important to an organization's success, driving a burgeoning need for industrial-strength sites with applications to match. Netscape's server-side JavaScript is one of the most powerful technologies available for meeting this need.

JavaScript is one of the most popular programming languages ever. The core language includes all of the statements, operators, objects, and functions that make up the basic JavaScript language and are applicable to both client and server applications. JavaScript has additional objects and functions specific to client-side or server-side functionality. You use client-side JavaScript, which is supported by Netscape Navigator, for working with Web pages and client browsers. Server-side JavaScript, available in Netscape Enterprise Server and Netscape's e-commerce products, enables back-end access to databases, file systems, and servers.

Why Server-Side JavaScript?

Although client-side JavaScript has been widely accepted and used in Web pages for some time now, developers are only beginning to understand the power and flexibility that server-side JavaScript offers, including a number of functions, objects, and structures for developing Web-based applications. Such applications can combine client-side and server- side JavaScript and HTML. One type of Web application, for example, may include a group of Web pages each serving a different function and accessed based on user selections or actions. Unlike traditional Web pages that rely solely on HTML, however, these pages can be generated entirely from JavaScript or a combination of HTML and JavaScript.

Client-side scripting has been very useful for creating dynamic content for Web pages and providing preliminary validation for user input,

but you often need a server component to handle the follow-on processing of information submitted by the client. For example, if a user submits a registration form, the server component could parse the form contents, set up an account, and then inform the user that the account is ready. Traditionally this has meant some sort of CGI solution. Server-side JavaScript is more powerful and easier to use than CGI, and it can dramatically reduce programming complexity and development time. You can often replace hundreds of lines of Perl or thousands of lines of C or C++ with a few dozen lines of HTML and JavaScript.

Server-side JavaScript's advantage stems from the fact that it has a well-developed framework for handling a wide range of common tasks. Because this framework contains high-level functions and objects, you can develop advanced solutions with minimal coding.

The Architecture

Most JavaScript applications follow a three-tier client/server architecture, with Web clients in the first tier, a Netscape Web server and database client in the second tier, and database servers and legacy data in the third tier. The first tier provides the user interface for the application, the necessary code to pass information to the second tier, and often data validation routines. As indicated earlier, the server passes HTML and client-side JavaScript to the client without interpreting them. Because of this, the type of Web client really doesn't matter as long as the client has a compatible JavaScript interpreter. Your needs will determine whether you tailor your front- end for a specific client or use generic functionality that will work in all JavaScript-capable clients.

The second tier provides the application logic and manages security for the application using server-side JavaScript. This tier consists of a Netscape HTTP server with JavaScript support and, if the application accesses a database server, a database client. The access controls on this tier determine whether clients outside a firewall can access the application.

The second tier is also where you'll find the Netscape server and JavaScript runtime environments. The key component of the Netscape server runtime is NSAPI, which connects your applications and the JavaScript runtime environment to the server. Key components of JavaScript's runtime environment include the JavaScript runtime library, the LiveWire database-access library, and the Java Virtual Machine. The JavaScript runtime library provides the primary server- side services, objects, and functions. LiveWire lets your server-side applications access relational databases. The Java Virtual Machine is used by Java applications running on the server.

The third tier supplies access to data and functions provided by database or application servers. You can also access legacy data and code written in programming languages such as C and C++. Normally, you do this through LiveConnect, an enabling technology that connects JavaScript with other technologies such as applets and plug-ins.

Now let's take a closer look at the features of server-side JavaScript, including session management, file and mail services, and database access. We'll also look at services for accessing other programming languages.

Session Management

Session Management service manages requests and maintains state information. The service consists of five objects: request, client, project, server, and Lock. To understand these objects, you need to understand how they are used and when they are created in the runtime environment.

Request is the shortest-lived of the session management objects,

lasting only for the duration of a single client request. Typically, client requests occur when a user enters a URL or clicks on a hyperlink. You can use request to examine information sent in client requests, such as the data submitted in a form, or to examine information about the client and the client request, such as the type of client being used.

To manage user sessions and maintain state information, you will rely on the client object. Client maintains user-specific information across multiple requests. Although many different users can access an application simultaneously, each user has a unique client object for that application. Further, users always have different client objects for different applications--even if those applications are running on the same server. You can use client to maintain user preferences, such as settings for colors and fonts, or users' account information, such as user names, passwords, and account numbers.

The runtime engine actually creates and destroys client objects with each request. By destroying client objects that are not currently in use, the runtime engine saves system resources--notably memory. You can't restore a client object at the beginning of each request without client-maintenance techniques such as cookies or URL-encoding.

Often you will need to pass information among clients and applications. You do this with the project and server objects. Project maintains state information for multiple clients using a particular application; server maintains state information for multiple applications running on a particular server. These objects are often used in conjunction with Lock, which lets you control access to a section of code temporarily.

You can use project to track any properties or values related to a single application. Application administrators may need to obtain current usage summaries, such as the current number of users or the average number of transactions per hour. Project can also handle real- time interactions among users, of the sort online chat programs require.

A new project object is created each time you start an application, and the object is destroyed any time you stop or restart an application. Because a single server can have multiple separate instances of an application, each instance of an application has its own unique project object.

Server objects let applications share information. For example, as a user moves from one application to another, server can pass on the user's preferences and login information. A server object is created when the server process is started and destroyed when the server process is stopped. All applications using the same server process share the same server object. Any time a host machine runs multiple server processes on different ports there will be multiple server objects. These objects will correspond to the specific server processes and ports they answer on. For example, a server process on port 80 will have a different server object than a server process on port 8080.

Netscape's server provides a multithreaded environment for JavaScript applications. Though multithreading speeds up processing and improves performance, it can lead to conflicts over which process has control of an application's objects and properties at any given time. To prevent this, an application should control access to the project and server objects. The application can handle access controls using either explicit locking or an instance of the Lock object.

With explicit locking, project and server are locked by invoking their respective Lock() methods. Lock() blocks access to the object until unlock() is called. Unfortunately, there is no time-out built in, so users can be locked out of the entire application if unlock() isn't invoked. As this isn't optimal, you may want to create an instance of the Lock object

instead. With Lock, you control not only access to a key section of code but also how long the lock lasts and which objects and properties it affects.

File and mail Services

File System Service lets you store persistent data and manipulate files. The File object is not intended for working with databases but is very useful when you need an easy-to-use data-storage solution. You use File to read, write, and search for files on the application server. You can't use File to work with files on the user's system.

File can work with both ASCII text and binary-formatted files, but reading and writing ASCII is much easier, because you can read and write data line-by-line or use strings. With binary files, on the other hand, you must break strings down into individual characters before you read or write. These individual characters are one byte in length and must be converted to a numeric value for reading and writing.

Mail Service consists of the SendMail object; you use it to generate mail messages within server-side JavaScript applications. The term SendMail may remind you of the Unix process used to generate mail messages, but the SendMail object is available for both Windows and Unix environments. SendMail lets you send multimedia such as images, sound files, and videos and also supports the multipart message syntax, which lets you send messages with attachments.

Once you create an instance of SendMail, you can use its properties to determine the fields used in your e-mail messages, how content is encoded, and where the message is sent. Generally, properties of SendMail correspond to actual fields in the e-mail message. For example, you use the To property to name recipients and the From property to name the sender.

SendMail directly supports the general fields used with standard SMTP servers, so you can create properties for custom fields as well, and then use these fields with custom mail servers or clients. Because custom fields are sent in the message header, they are generally subject to the restrictions of RFC 822, the specification that defines the standard for Internet messages (which states that all user-defined fields should have the prefix X-).

Database Access

Server-side JavaScript provides a complete Web-to-database solution, and whether you want to create a database-query engine or a complete database-management system, you can use server-side JavaScript's LiveWire Database Service to help get the job done. The service supports native drivers for DB2, Informix, Oracle, and Sybase, as well as for databases that conform to the ODBC standard. Features supported by the service include database pooling, pass-through SQL, cursors, and stored procedures.

Keep in mind though that your server environment ultimately determines the types of databases you can work with, as well as the features of LiveWire Database Service that are available to you. So before you begin any project planning or work on a Web-to-database application, you should determine whether the service works within your current server environment. If the service does not support your current server environment, you may need to modify or upgrade your HTTP server, database client, or ODBC components. And even if the service works, you'll still need to make sure that it supports the features you need.

There are four basic steps for working with a database: mapping the database structure, establishing a connection to the database, handling database transactions as necessary, and finally, releasing the connection. Let's explore these steps in more depth.

Map out the database structure. To extract records from a database, you may need to know the names of tables and columns. To update data, you

may need to know the acceptable data types, the maximum allowable data size, whether the column accepts null values, and other pertinent information. If you are working with an existing database, obtaining this information before coding your application is fairly easy. But if you're creating a new database for the application, you should create a prototype of the database before you start programming the application. Once the database structure is in place, map out its properties, writing down the pertinent information such as table names, column names, and acceptable data types. This makes working with the database much easier.

Establish a connection. LiveWire Database Service provides two ways to connect to a database: shared connections and pooling. A shared connection is accessible to a single user at a time; other users are locked out of the database as long as that user is connected. With pooling, on the other hand, multiple users can access a database at the same time via a connection pool.

Think of a connection pool as a jar full of access cards. When a user needs access to a database, he takes a card out of the jar. When others need access to the database, they can take cards out of the jar, and so on. If at any time the jar becomes empty, the next user must wait until a card is returned to the jar.

With either connection method, application performance is often tied directly to the availability of database connections. If no connection is available when a user needs one, the application has to try to obtain one and the user has to wait. Because of this, database connections are usually established on the application's initial page and then made available to users on an as-needed basis. The exception is when your application requires users to log on to the database with a user-specific sequence. In this case, you would probably want to establish the connection when it's needed. For example, if users must access the database using their login names and passwords, you should establish connections the first time users try to access the database.

Handle transactions. Once you establish a connection, users can perform any actions permitted by their profile, which may include database queries and updates. Database queries can be handled with pass-through SQL, read-only cursors, and stored procedures. Database updates can be handled with pass-through SQL, updateable cursors, and stored procedures.

Whenever you update a database, you should use explicit transaction controls. With transactions, you perform a group of actions together. If the actions are successful, you update the database. If any of the actions are unsuccessful, you restore the database to its original state.

Release the connection. When a user completes a transaction, you usually want to release the database connection. Releasing the connection doesn't necessarily mean breaking the connection with the database. With a shared connection, you release the lock on a database object, which ensures that other users can access the database. With a pooled connection, you return the connection to the pool, which ensures that it is available to other users.

Accessing Other Languages

If you have existing functions written in another programming language and you want to incorporate their functionality into a JavaScript application, JavaScript's external library framework will come in handy. Using this framework, JavaScript applications can call functions written in C--and through C, other programming languages, such as C++ or Visual C++. Because you don't have to convert the functions to JavaScript, you reduce development time and often gain the advantage of improved performance.

Functions written in a programming language other than JavaScript are called native functions and must be placed in an external library that can

be accessed or shared dynamically in the runtime environment. For Windows systems, the external library must be a dynamic link library (DLL). For Unix systems, the external library must be a shared object. Once you place a native function in an external library, you must register the library with the JavaScript runtime environment. Registering the library tells the runtime engine that applications are cleared to access the library.

If you want to go beyond the external library framework, you'll need to use LiveConnect. In the server-side environment, LiveConnect is essentially a set of extensions for Java. These extensions make it possible for JavaScript to access Java, and through Java any objects that are CORBA-compliant. CORBA (short for Common Object Request Broker Architecture) specifies an architecture for communication between distributed objects. With CORBA, the operating system and the object's programming language don't matter as long as the object complies with the architecture.

Java objects interact with other objects via an intermediary called an ORB (object request broker), which lets objects make and receive requests from other objects. When a request comes in, the ORB intercepts it and maps the request to the object that is supposed to handle it. Afterward the ORB invokes the appropriate method in the receiving object and passes available parameters as necessary. Finally, it returns the results to the object that made the request. Fortunately, once you create the necessary objects and extend them for CORBA, all this interaction takes place behind the scenes.

Server-side JavaScript lets you leverage your knowledge of this powerful scripting language to create dynamic Web applications that can run in any browser and on any platform. It's especially useful for building the database-driven applications so necessary to corporate success on the Web today.

William Robert Stanek is a frequent contributor to PC Magazine. He can be contacted at writing@tvpress.com.

COMPANY NAMES: Netscape Communications Corp.--Innovations
GEOGRAPHIC CODES/NAMES: 1USA United States
DESCRIPTORS: Java; Programming tutorial
EVENT CODES/NAMES: 390 Nonmanufacturing technology
PRODUCT/INDUSTRY NAMES: 7372422 (DBMS Utilities)
SIC CODES: 7372 Prepackaged software
NAICS CODES: 51121 Software Publishers
TICKER SYMBOLS: NSCP
FILE SEGMENT: CD File 275

ordered

00525217 99MY02-001

Creating a Sherlock module to search your Web site

The MacAuthority , February 1, 1999 , v8 n2 p1-4, 4 Page(s)
ISSN: 1062-452X
Company Name: Apple Computer
URL: http://www.apple.com/sherlock http://www.apple-donuts.com
Product Name: Sherlock
Languages: English
Document Type: Articles, News & Columns
Geographic Location: United States

Discusses Sherlock, the Mac OS 8.5 software's new Internet search utility. Says that Apple **designed** Sherlock with a plug-in architecture that allows users to **create** their own **search modules** for their favorite search engines - even their own Web sites. States that creating a Sherlock module for a Web site is not as difficult as first imagined. Notes that you need a good understanding of HTML, ResEdit, and a text editor; from there the function of your search engine must be analyzed and the same function must be built into the module. Concludes that if you need assistance with Sherlock, there are several sites on the Internet to find help - Apple's Web site and the Sherlock Internet Search Archives. Includes six screen displays.

Descriptors: Search Engines; Internet; Plug-ins; Web Sites; HTML
Identifiers: Sherlock; Apple Computer